Assignment 11 Advanced Hbase

Task 1

Explain the below concepts with an example in brief.

● *Nosql Databases*

NoSQL is an approach to database design that can accomodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

Ex Cassandra , Mongo DB, HBASE

● Types of Nosql Databases

1)Key-value stores

Key-value stores, or key-value databases, implement a simple data model that pairs a unique key with an associated value. Because this model is simple, it can lead to the development of key-value databases, which are extremely performant and highly scalable for session management and caching in web applications. Implementations differ in the way they are oriented to work with RAM, solid-state drives or disk drives. Examples include Aerospike, Berkeley DB, MemchacheDB, Redis and Riak.

2)Document databases

Document databases, also called document stores, store semi-structured data and descriptions of that data in document format. They allow developers to create and update programs without needing to reference master schema. Use of document databases has increased along with use of JavaScript and the JavaScript Object Notation (JSON), a data interchange format that has gained wide currency among web application developers, although XML and other data formats can be used as well. Document databases are used for content management and mobile application data handling. Couchbase Server, CouchDB, DocumentDB, MarkLogic and MongoDB are examples of document databases.

3)Wide-column stores

Wide-column stores organize data tables as columns instead of as rows. Wide-column stores can be found both in SQL and NoSQL databases. Wide-column stores can query large data volumes faster than conventional relational databases. A wide-column data store can be used for recommendation engines, catalogs, fraud detection and other types of data processing. Google BigTable, Cassandra and HBase are examples of wide-column stores.

4)Graph stores

Graph data stores organize data as nodes, which are like records in a relational database, and edges, which represent connections between nodes. Because the graph system stores the relationship between nodes, it can support richer representations of data relationships. Also, unlike relational models reliant on strict schemas, the graph data model can evolve over time and use. Graph databases are applied in systems that must map relationships, such as reservation systems or customer relationship management. Examples of graph databases include AllegroGraph, IBM Graph, Neo4j and Titan.

● *CAP Theorem*

1)Consistency - This means that the data in the database remains consistent after the execution of an operation. For
example after an update operation, all clients see the same data.

2)Availability - This means that the system is always on (service guarantee availability), no downtime.

3)Partition Tolerance - This means that the system continues to function even if the communication among the servers is
unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

● ***HBase Architecture***

HBase is composed of three types of servers in a master slave type of architecture.
→ Region servers serve data for reads and writes
→HBase Master process handles the Region assignment, DDL (create, delete tables) operations
→Zookeeper maintains a live cluster state
The Hadoop DataNode stores the data that the Region Server is managing
→All HBase data is stored in HDFS files
→The NameNode maintains metadata information for all the physical data blocks that comprise the files

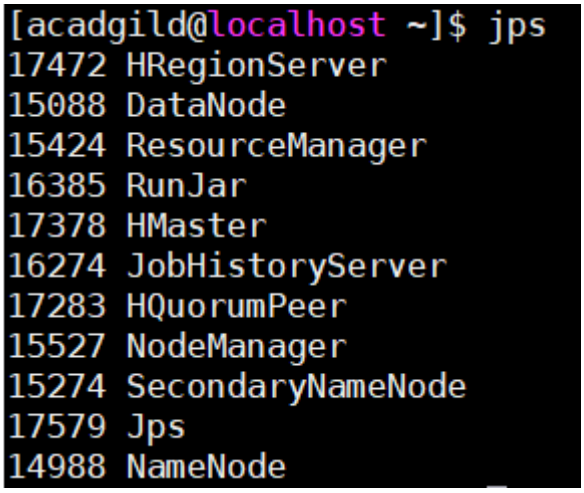
● ***HBase vs RDBMS***

HBASE	*RDBMS*
It is distributed, column oriented, versioned data storage system.	It is designed to follow FIXED schema. It is row-oriented databases and doesn't natively scale to distributed storage.
HDFS is underlying layer of HBase and provides fault tolerance and linear scalability. It doesn't support secondary indexes and support data in key-value pair.	It supports secondary indexes and improvises data retrieval through SQL language.
It supports dynamic addition of column in table schema. It is not relational database like RDBMS.	It has slow learning curve and support complex joins and aggregate functions.
HBASE helps Hadoop overcome the challenges in random read and write.	

Task 2

ImportTSV Data from HDFS into HBase

First we need to start Hadoop daemons, job history server and hbase.



Creating a table with two column family

```
hbase(main):002:0> create 'bulktable','cf1','cf2'
0 row(s) in 1.2520 seconds

=> Hbase::Table - bulktable
hbase(main):003:0>
```

Creating a file

```
[acadgild@localhost Hbase]$ pwd
/home/acadgild/Hbase
[acadgild@localhost Hbase]$ ll
total 4
-rw-rw-r--. 1 acadgild acadgild 40 Dec  2 08:10 bulk_data.tsv
[acadgild@localhost Hbase]$ cat bulk_data.tsv
1       Amit    4
2       Girija  3
3       Jatin   5
4       Swati   3
[acadgild@localhost Hbase]$
```

Putting the file into HDFS and importing into HDFS with importTsv command

Hadoop dfs -put bulk_data.tsv /hbase

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:name,cf2:exp bulktable /hbase/bulk_data.tsv

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hbase]$ hdfs dfs -cat /hbase/bulk_data.tsv
18/12/02 08:43:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
1       Amit    4
2       Girija  3
3       Jatin   5
4       Swati   3
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Hbase]$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:name,cf2:exp bulktable /hbase/bulk_data.tsv
2018-12-02 08:49:54,014 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicab
le
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2018-12-02 08:49:55,103 INFO  [main] zookeeper.RecoverableZooKeeper: Process identifier=hconnection-0x6025e1b6 connecting to ZooKeeper ensemble=localhost:2181
2018-12-02 08:49:55,117 INFO  [main] zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.6-1569965, built on 02/20/2014 09:09 GMT
2018-12-02 08:49:55,117 INFO  [main] zookeeper.ZooKeeper: Client environment:host.name=localhost
2018-12-02 08:49:55,117 INFO  [main] zookeeper.ZooKeeper: Client environment:java.version=1.8.0_151
2018-12-02 08:49:55,117 INFO  [main] zookeeper.ZooKeeper: Client environment:java.vendor=Oracle Corporation
```

Data in the table

```
hbase(main):002:0> scan 'bulktable'
ROW                                   COLUMN+CELL
0 row(s) in 0.0720 seconds

hbase(main):003:0> scan 'bulktable'
ROW                                   COLUMN+CELL
 1                                    column=cf1:name, timestamp=1543720793908, value=Amit
 1                                    column=cf2:exp, timestamp=1543720793908, value=4
 2                                    column=cf1:name, timestamp=1543720793908, value=Girija
 2                                    column=cf2:exp, timestamp=1543720793908, value=3
 3                                    column=cf1:name, timestamp=1543720793908, value=Jatin
 3                                    column=cf2:exp, timestamp=1543720793908, value=5
 4                                    column=cf1:name, timestamp=1543720793908, value=Swati
 4                                    column=cf2:exp, timestamp=1543720793908, value=3
4 row(s) in 0.1690 seconds

hbase(main):004:0>
```